Format Specification Shapes (.json)

Shape files are created in JSON format and are therefore subject to the requirements for this format.

A separate Shapes file should always be created and maintained for each model or series of a supplier.

Shape files consist of the following objects on the first level:

- Scheme (optional),
- General SVG settings (#svg),
- General furniture settings (#common) und
- List of all symbols (Shapes).

For all length values: format is **Number**, the decimal separator is a **.** and **m** is expected as unit.

This documentation requires a version 1.2.5 or higher of IG.GFX.ShapeCreator.

Scheme

In the JSON the Shapes JSON Schema file can/should be specified in the first place. This allows intelligent editors like Visual Studio Code to provide parameter validity and auto-completion when creating the file.

```
"$schema": "https://archive.intelligentgraphics.biz/schemas/gfx/shapes.json"
```

General SVG settings

In the JSON file, the object **#svg** is used to provide the IG.GFX.ShapeCreator with general settings for the SVG creation. The following properties are supported:

Name	Type	Default	Description/Values
Scale	Number	0.1	The scaling for the SVG. Default is 0.1.
OffsetX	Int	0	Increases the ViewBox of the SVG (width) by n pixels, recommended is 1.
OffsetY	Int	0	Increases the ViewBox of the SVG (height) by n pixels, recommended is 1.
AttachPoints	Boolean	false	Creates leader lines for all symbols for use in XcR-Cat if **true**.
AttachMode	String	LR	Creates, if **AttachPoints** is true, the left (L) and/or right (R) attachment line. With (C) a mode is activated that creates two attach lines, which can be useful in opening corners.
AttachOrientation	String	left	The attachment lines created by the mode (AttachMode = "C") will be aligned to the left of the origin if **left** is specified, if **right** is specified they will be aligned to the upper right side.

Attachment lines should only be created if you are dealing with single elements that can be planned. You should not create any lines for composite elements.

Information and examples on attachment lines follow in a later section.

Inheritance

All SVG settings apply to all symbols, but they can be overridden per symbol if necessary.

Example

```
"#svg": {
    ...
    "AttachPoints": false,
    "AttachMode": "LR"
},
"#common": { ... },
"Shapes": {
        "Armrest_Right": {
            "#svg": {
                "AttachPoints": true,
                "AttachMode": "L"
            },
            "amrest": {
                "#armrest#1": {}
            }
        },
...
}
```

Additional SVG settings on symbol level

As mentioned above, all SVG settings are inherited by all child elements.

In addition, the following properties can be assigned:

Name	Typ	Default	Description/Values
Product	String		Name of the product from the commercial data if the IG.GFX.ShapeCreator is to create a list of attachment points. Information about this can be found in the documentation of the IG.GFX.ShapeCreator.
AttachLeftRotation	Int	0	The left attachment line, if specified, is rotated by 90 or -90°.
AttachLeftOrigin	String	origin	If origin is set, the left attachment line is created at the origin, if size is specified, the join line starts at the depth of the element (total depth).
AttachRightRotation	Int	0	The right attachment line, if specified, is rotated by 90 or -90°.
AttachRightOrigin	String	origin	If origin is set, the right attachment line is created at the origin, if size is specified, the join line starts at the depth of the element (total depth).

General furniture settings

In the JSON file, the object **#common** is used to provide the IG.GFX.ShapeCreator with general values for the furniture to be created for the SVG creation. The following properties are supported:

Property	Type	Default	Description/Values
Mode	String	cushion	The type of furniture, controls the display of shapes. Currently only upholstered furniture (**cushion**) is supported.
ArmrestWidth	Number	0.15	Specifies the width of the armrests (**cushion** mode).
ArmrestDepth	Number	1.0	Specifies the depth of the armrests (**cushion** mode).
BackseatDepth	Number	0.15	Specifies the depth of the back cushions (**cushion** mode).
SeatDepth	Number	0.85	Specifies the seating depth of the sofas (**cushion** mode).
Rounding	Number	0.05	Specifies the standard diameter of the rounded corners.
Model	String		Name or Id of the model/series for the creation of the symbols.

Notes

- ArmrestWidth should always be specified in the **#common** object. If the value is not known, **0.15** should be used.
- ArmrestWidth is always included in the calculation of the total width, so when determining the width of a couch, the value should be taken into account.
- ArmrestDepth should always be specified in the **#common** object. If the value is not known, **1.00** should be used.
- BackseatDepth should always be specified in the **#common** object. If the value is not known, **0.15** should be used.
- SeatDepth should always be specified in the **#common** object. If the value is not known, **0.85** should be used.
- SeatDepth plus BackseatDepth gives the total depth of the couch.

Example

```
"#common": {
    "Mode": "cushion",
    "ArmrestWidth": 0.15,
    "ArmrestDepth": 0.72,
    "BackseatDepth": 0.15,
    "SeatDepth": 0.57,
    "Rounding": 0.05,
    "Model": "test"
}
```

Symbols

For each product of a model or series for which a 2D symbol is required, a symbol should be defined in the Shapes object. The name of the object also specifies the file name, so you should use meaningful product names. Characters outside the alphabet and numbers as well as _ or - should be avoided. Spaces are to be avoided in any case.

Each symbol in turn consists of a list of **components**, which are to be understood like a construction plan. Each component in turn can consist of a number of defined **character elements**.

Parts and drawing elements in turn can have defined properties that influence the generation of SVG symbols.

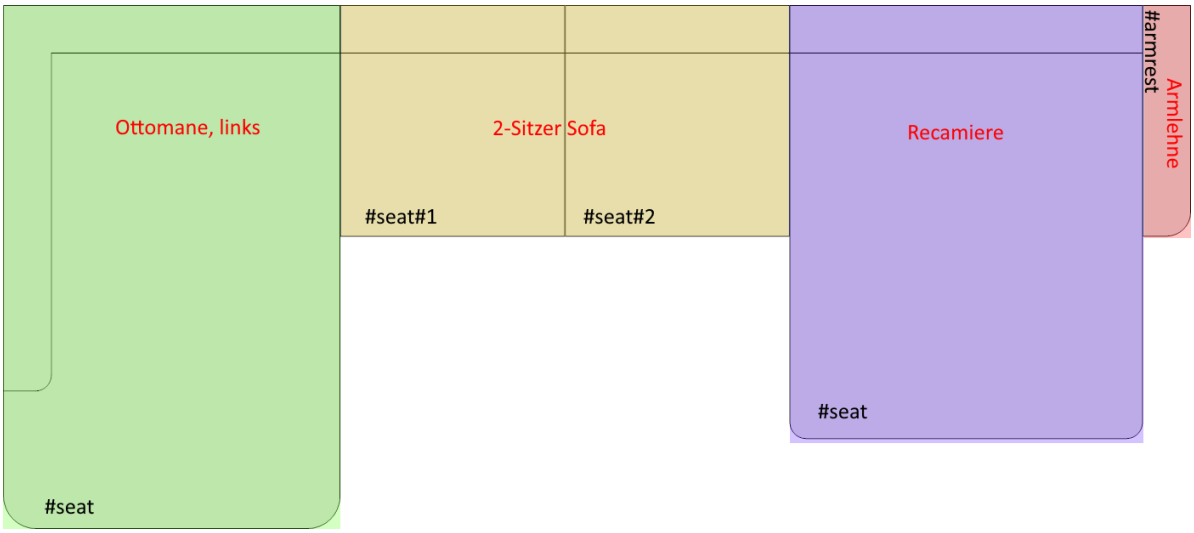


Figure: Components (red text) and associated drawing elements (black text).

This is best illustrated by the following examples.

Example 1, Stool

```
"Hocker_100x100": {
      "hocker": {
        "Width": 1.0,
        "Depth": 1.0,
        "#box": {
          "corner": "round"
        }
      }
}
```

A stool usually consists of exactly one component, here **"stool "**. The width and depth are specified as 1.0 m. The defined drawing element **#box** is used for the representation, here with round corners. The diameter of the corners is taken from the general furniture settings (#common, Rounding).

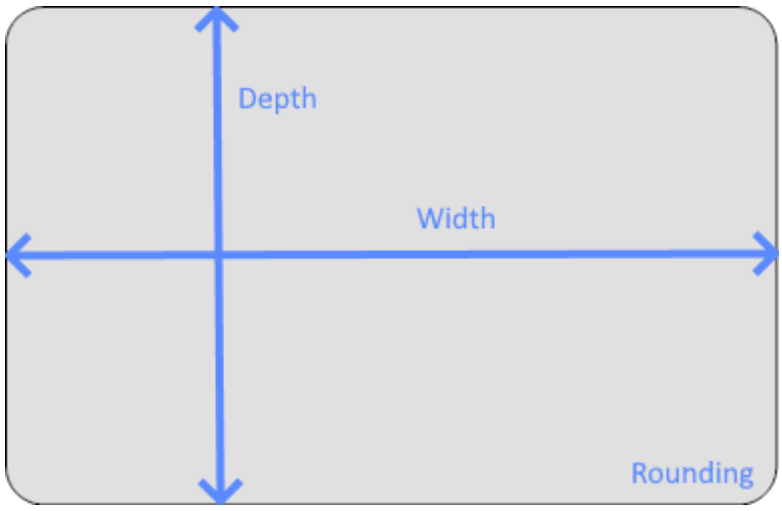


Figure: Stool with character element #box and the properties

Width and **Depth** are properties of the component **"stool "**, corner is a property of the defined drawing element **#box**.

Example 2, Combination of 2-seater and Recamier

```
"2SEATER_REC_RECHTS": {
      "2": {
        "Width": 1.4,
        "#armrest#1": {
          "corner_3": "round",
          "diameter_3": 0.075
        },
        "#seat#1": {},
        "#seat#2": {}
      },
      "rec": {
        "Width": 1.1,
        "#seat#1": {
          "SeatDepth": 1.2,
          "corner_2": "round",
          "diameter_2": 0.05,
          "corner_3": "round",
          "diameter_3": 0.05
        },
        "#armrest#1": {
          "corner_2": "round",
          "diameter_2": 0.075
        }
      }
}
```

This combination has a total width of 2.8 m and a total depth of 1.35 m (recamiere). The sofa itself is 0.72 m deep with a back section depth of 0.15 m.

The combination consists of the components "2" (2-seater sofa with armrest left) and "rec" (recamier with armrest right). The sofa has a total width of 1.55 m (incl. armrest) and the recamier has a total width of 1.25 m (incl. armrest).

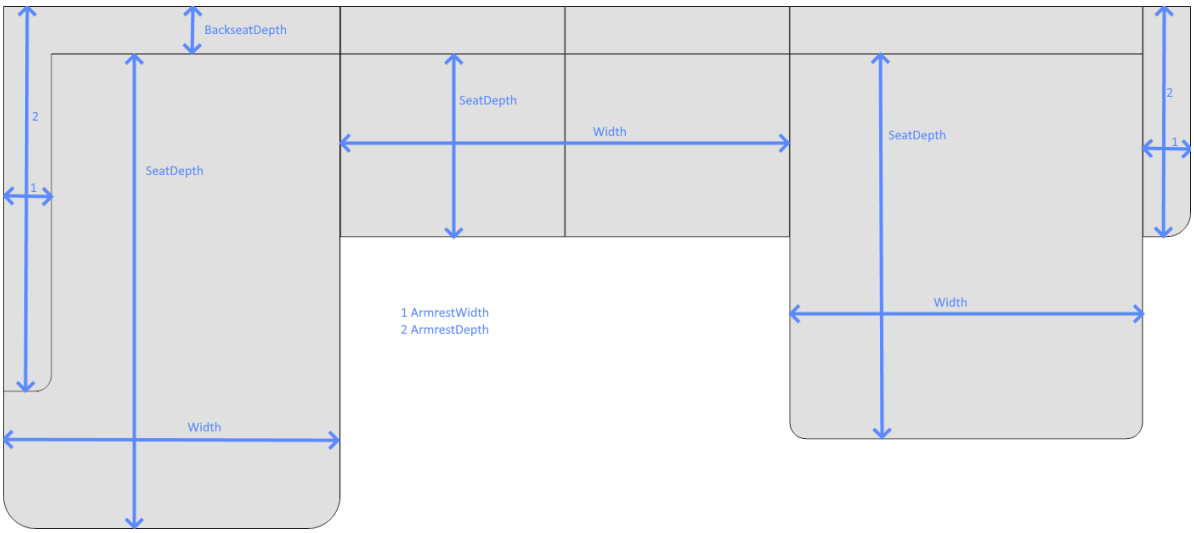


Figure: Combination of several components and elements and the properties (exemplary).

For the representation of the sofa and the recamier the drawing elements **#armrest** and **#seat** are used. Depending on the requirements, these are adjusted via supported properties.

Example 3, Combination with corner

The following example shows a corner combination.

```
"2_e_3": {
      "2": {
        "Width": 1.4,
        "Rotation": -90,
        "#armrest#1": {
          "corner_3": "round",
          "diameter_3": 0.075
        },
        "#seat#1": {},
        "#seat#2": {}
      },
      "e": {
        "#corner#1": {
          "Width": 0.83,
          "Depth": 0.83,
          "Orientation": "left"
        }
      },
      "3": {
        "Width": 2.1,
        "#seat#1": {},
        "#seat#2": {},
        "#seat#13": {},
        "#armrest#1": {
          "corner_2": "round",
          "diameter_2": 0.075
        }
      }
}
```

For corners it is important to rotate the corresponding parts before corner using the **Rotation** property. In this case it is left aligned corner with a rotation of -90°, for right aligned corners the rotation must be set to 90°.

Component properties

Components support the following features:

Property	Type	Default	Description/Values
Width	Number	1.0	Specifies the total width of the component.
Depth	Number	1.0	Specifies the total depth of the component.
Rotation	Grad	0	Rotates the entire component (including the elements) by n degrees.
Orientation	String	none	Aligns the component. if supported. Possible values are **none**, **left** or **right**.
ArmrestWidth	Number	0.15	Specifies the width of the armrests (**cushion** mode).
ArmrestDepth	Number	1.0	Specifies the depth of the armrests (**cushion** mode).
BackseatDepth	Number	0.15	Specifies the depth of the back cushions (**cushion** mode).
SeatDepth	Number	0.85	Specifies the seating depth of the sofas (**cushion** mode).

Inheritance

All parts and drawing elements use the values defined in the general #common settings. However, these can be overridden per part (or drawing element) if required, as indicated in the table above.

Properties of drawing elements

- The following drawing elements are defined in the "**cushion**" mode and can be used in the parts:
  - #box
    A box with width and height and optional round corners
  - #seat
    A seat with backrest
    With the property "**RenderMode**" the rendering can be adapted for different purposes, see overview below
  - #armrest
    An armrest with optional rounded corners, see overview below
  - #corner
    Corners are special drawing elements to be able to plan combinations around corners

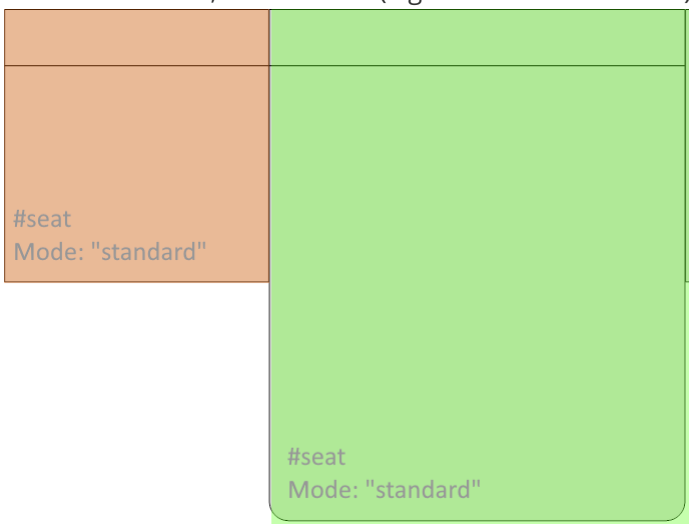
  Each drawing element can have the following properties:

Property	Type	Default	Description/Values
Width	Number	1.0	Set width for the drawing element.
Depth	Number	1.0	Set depth for the drawing element.
ArmrestWidth	Number	0.15	Specifies the width of the armrests (**cushion** mode).
ArmrestDepth	Number	1.0	Specifies the depth of the armrests (**cushion** mode).
BackseatDepth	Number	0.15	Specifies the depth of the back cushions (**cushion** mode).
SeatDepth	Number	0.85	Specifies the seating depth of the sofas (**cushion** mode).
Orientation	String	none	Aligns the character element. if supported. Possible values are **none**, **left** or **right**.
RenderMode	String	standard	Changes the appearance of the created seat, corner and or other basic types.
corner	String	corner	Changes the display of all corners, supported is **corner** (Corners) or **round** (Rounded corners). Only for drawing element **#box**!
diameter	Number	0.05	Specifies the diameter for the rounded corners. Only for drawing element **#box**!
corner_2	String	corner	If "round" the right lower corner of the seat will be rounded. Only for character elements **#seat** and **#armrest**!
diameter_2	Number	0.05	Specifies the diameter for the rounded corner.
corner_3	String	corner	If "round" the left bottom corner of the seat will be rounded. Only for character elements **#seat** and **#armrest**!
diameter_3	Number	0.05	Specifies the diameter for the rounded corner.

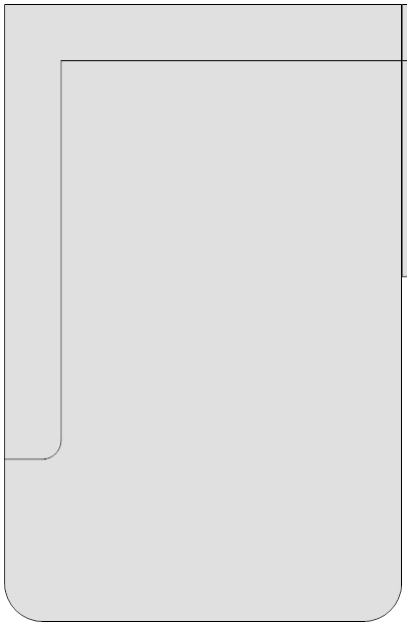
RenderMode (Drawing element #seat)

The element #seat can take the following representations:

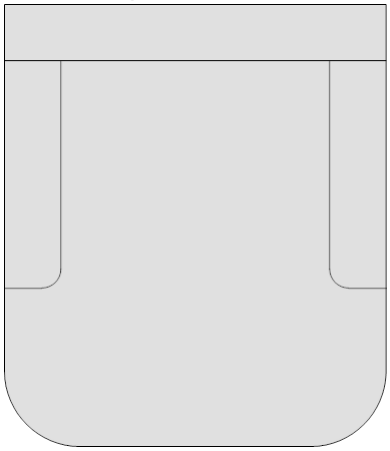
- Mode "standard", Normal seat (e.g. n-seater or recamier)



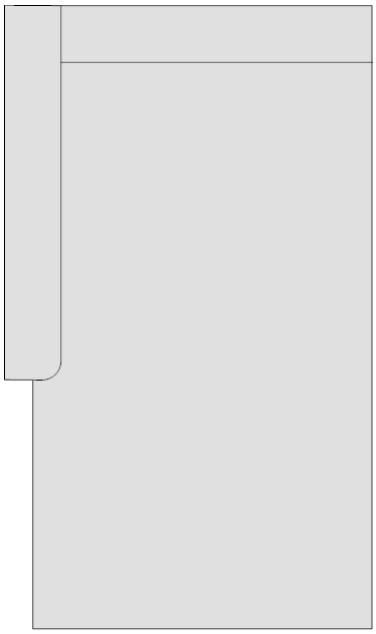
- Mode "ottomane", for ottomans



- Mode "single", for armchairs



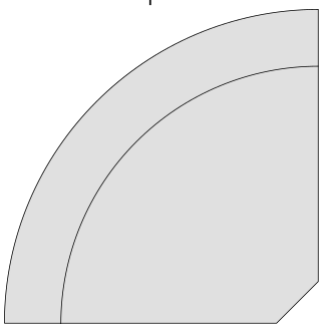
- Mode "divan", for divans



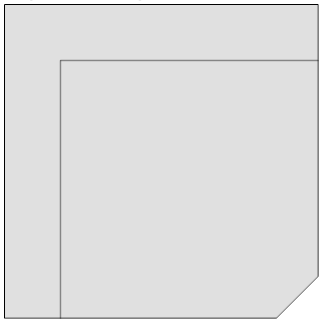
RenderMode (Drawing element #corner)

The element #corner can take the following representations:

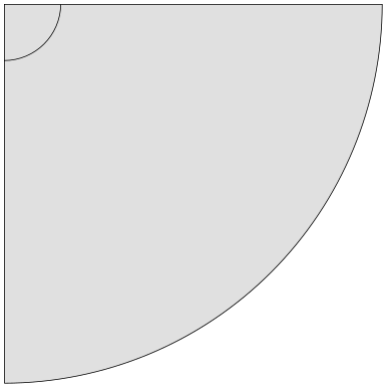
- Standard representation



- Square ("square") mode for angular corners



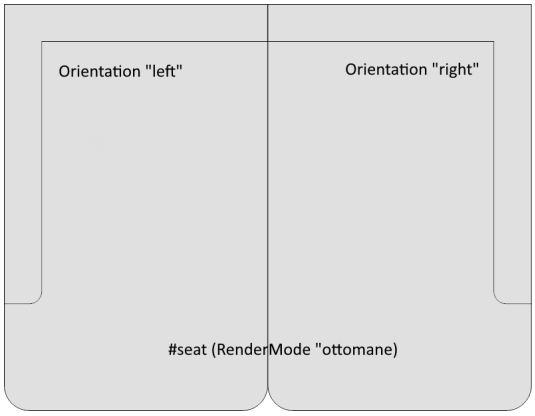
- Mode "roundMirror" for 270° corners, round display



Orientation (Orientation)

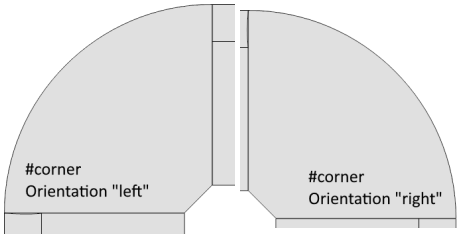
The following drawing elements support different orientations:

- Ottomans (#seat plus RenderMode "ottomane")



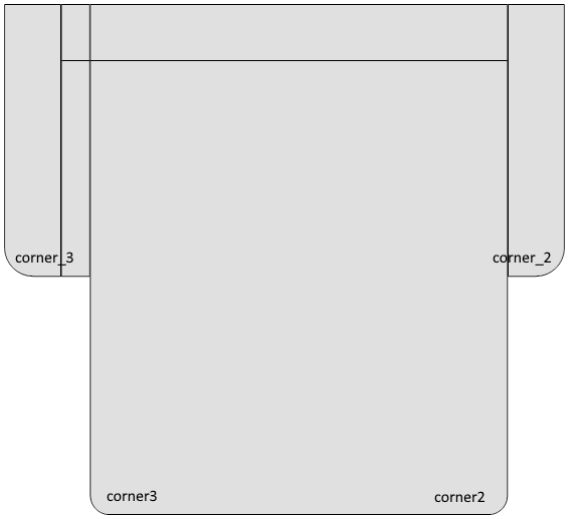
  Ottomans with orientation "left" are finishing elements on the left side and vice verca.

- Corners (#corner)



Rounding corners (Drawing element #seat und #armrest)

Seats and armrests support the rounding of the under left (3) and right corner (2).



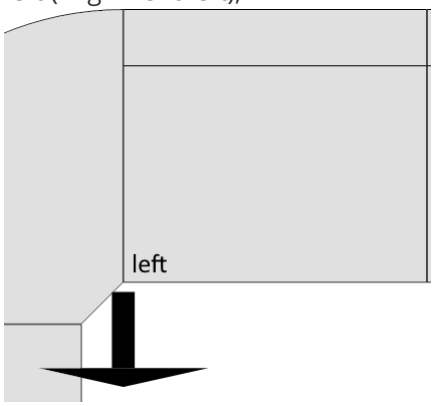
Extensions

Drawing elements can be given various extensions that can be used to display additional functions. These include, for example, a pull-out function, LED lighting or electronic seat depth adjustments.

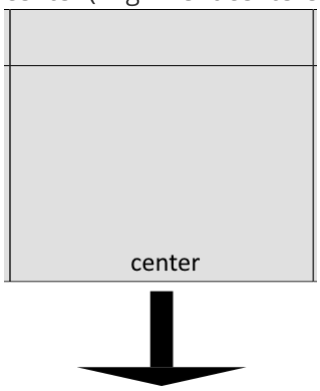
PullOut

A PullOut element represents an arrow for displaying a pull-out function. With the help of the "Align" property, the arrow can be aligned to the associated drawing element. Thereby

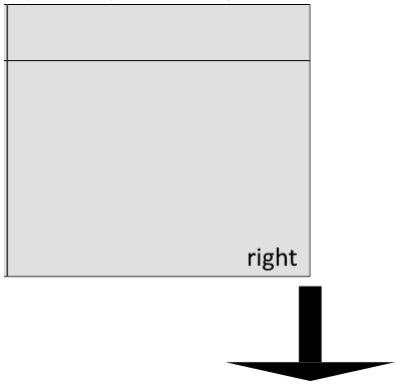
- left (Alignment left),



- center (Alignment centered) and



- right (Alignment right)



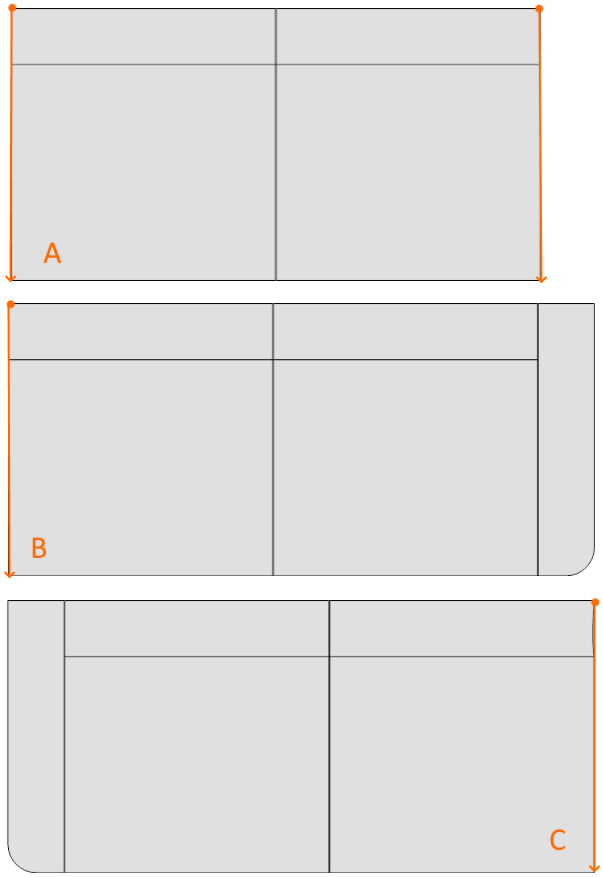
Attachment lines in the #svg element.

The "**AttachMode**" property tells IG.GFX.ShapeCreator which attachment lines to create.

Example

```
"#svg": {
"AttachPoints": true,
"AttachMode": "LR"
}
```

The following graphic shows the standard modes "LR" (A), "L" (B) and "R" (C).



Origin

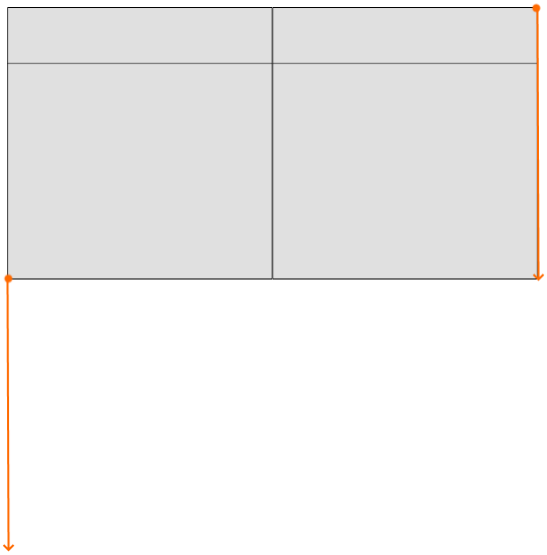
The origin of the attach lines in the "L" and "R" mode can be moved. For this purpose, the properties "**AttachLeftOrigin**" and "**AttachRightOrigin**" can be used and set to the value size. The default value is origin, but this need not be explicitly specified.

This shift is useful for corner elements, for example, to be able to define the correct attachment point.

Example

```
"#svg": {
"AttachPoints": true,
"AttachMode": "LR",
"AttachLeftOrigin": "size"
}
```

The following graph shows the shifted origin of the join lines when Origin is specified as "size".



Note: The length of the lines always results from the total depth of the created shape.

Rotation

It may be necessary to rotate the attach lines if elements have to be attached in a different direction. For this purpose the properties "**AttachLeftRotation**" and "**AttachRightRotation**" can be used.

The following examples and the corresponding figure show the possible values for the rotation.

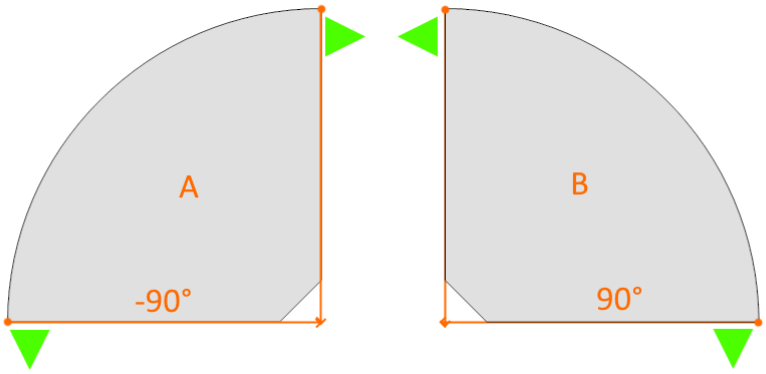
Variant **A**, the left attachment line is moved at the origin and then rotated to point "right".

```
"#svg": {
  "AttachPoints": true,
  "AttachMode": "LR",
  "AttachLeftRotation": -90,
  "AttachLeftOrigin": "size"
 }
```

Variant **B**, the right attachment line is moved at the origin and then rotated to point "left".

```
"#svg": {
  "AttachPoints": true,
  "AttachMode": "LR",
  "AttachRightRotation": 90,
  "AttachRightOrigin": "size"
 }
```

The figure shows the alignment and rotation of the joining lines as well as the imaginary joining directions.



Opening corners

Under certain circumstances corners can also have an angle of 270°, this circumstance is supported by IG.GFX.ShapeCreator with the value "C" in the feature "**AttachMode**". In this case, two matching attachment lines are created automatically and do not need to be rotated additionally. Both attachment lines have a common origin. This can be set to the left (value "left", default) or right upper corner (value "right") with the property "**AttachOrientation**".

In example A a left alignment is used, in example B a right alignment.

Note: in mode "C" the origin **cannot** be moved as in modes "L", "R" or "LR".

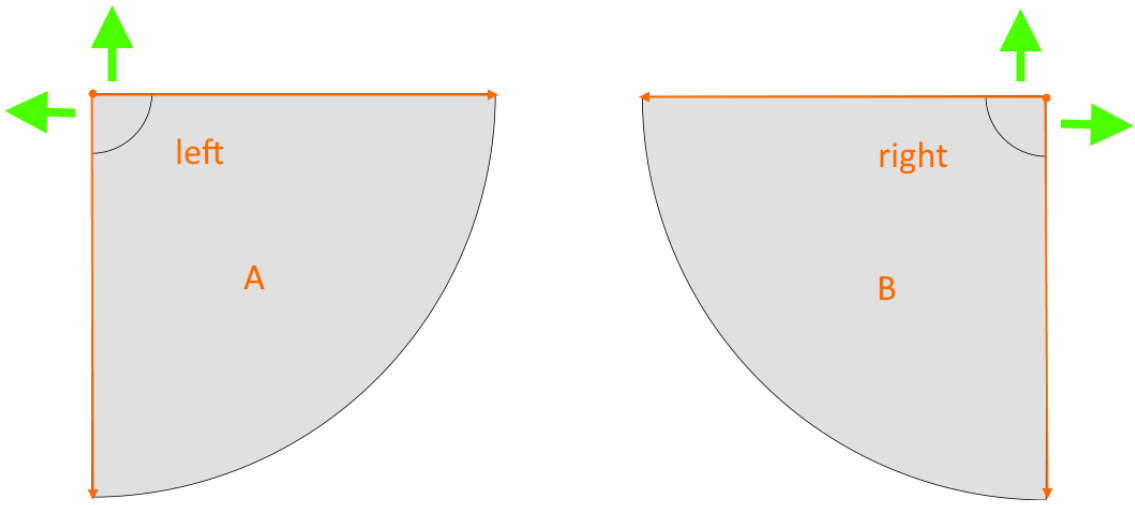
Example A

```
"#svg": {
  "AttachPoints": true,
  "AttachMode": "C",
  "AttachOrientation": "left"
 }
```

Example B

```
"#svg": {
  "AttachPoints": true,
  "AttachMode": "C",
  "AttachOrientation": "right"
 }
```

The following image shows the result of the properties in the #svg object.


left
A
right
B